1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

CHINESE TEXTUAL CLASSIFICATION BASED ON MULTILAYER FEED FORWARD WITH BACK PROPAGATION NEURAL NETWORK

Tyne Liang and Shyang-Tay Tseng

Department of Computer and Information Science,

National Chiao-Tung University, Hsinchu, Taiwan, R. O. C.

Email: {tliang, twoby}@cis.nctu.edu.tw

ABSTRACT

Due to the improvement of network technologies, development of text classification methods becomes urgent so as to provide an efficient retrieval in an increasing growth of electronic documents. In this paper, textual classifications based on a supervised neural network are investigated. Since the performance of a neural-based classifier is affected with selection of appropriate textual descriptors, two extraction methods are proposed namely, descriptor extending process and appending process. Meanwhile a parallel classifier is implemented in order to deal with overfitting problem. The performance of proposed models are verified with real textual data. Experimental results show that the parallel model is superior to simple multi-layer feed-forward with back propagation model and the model proposed by Kwok in terms of classification accuracy. Besides, both extending and appending processes indeed improve classification accuracy and speedup training time when they are implemented to different network classifiers.

1. INTRODUCTION

Textual classification reduces search space when an end user enquires a huge textual database. There are two classification approaches, traditional and neural-based classification models. Traditional textual classification models, such as vector space and probabilistic models, are based on similarity computation between classes and documents. Essentially vector space model is equivalent to probabilistic model [3][4]. Though vector space model is simple and easy for implementation, it has disadvantages of the assumed independence between keywords, and the lack of theoretical justification for an appropriate similarity function [14].

On the other hand, neural-based models implement classification with a feed-forward network and the network is built as a parallel architecture with specified learning algorithm. Multi-layer feed-forward (MLFF) neural network with back propagation (BP) learning algorithm is a common supervised neural network and proved to be suitable for classification [5].

Generally, a text can be viewed as a monolithic entity described by a list of terms. When occurrence frequency of a term within a text is ignored, each term can be treated with Boolean values. In early age, traditional models are implemented with Boolean vectors [2][9][10]. Relevance feedback was added to these models to improve classification accuracy [13][17]. Hierarchical clustering methods were employed to improve retrieving speed [6][11].

Contrast to the above models that treat a text as an monolithic entity, another kind of probabilistic model based on component theory treats a text as many non-monolithic but composed of independent components (such as sentences, phrases, or single terms) [7]. Though each component is also described by a list of terms, this model was added with self-learning capability and was implemented as a feed-forward neural network [8]. Meanwhile, a vector space model could be implemented as a simple three-layer neural network based on cosine similarity function [16]. Although these network models are similar to conventional neural-based models, they can append or delete classes easily and the weights among links are initialized by a specified weighting function [8][16]. On the other hand, neural-based models require new training process if classes are modified. Hence such approaches are suitable only for static environments.

Ng [12] used perceptron neural network to classify huge amount of news texts. Horng [1] used MLFF+BP and LVQ (Learning Vector Quantization) neural models with unweighted textual vectors and his experimental results indicate that neural-based textual classification is better than vector space and probabilistic models in terms of classification accuracy.

Though neural network approaches have capabilities of mapping, fault-tolerance, learning, and parallel processing, there is a problem with simple neural-based classifier. This problem occurs when the classifier deals with many number of classes, it will take long learning time. Such problem is called overfitting. In this paper, a parallel MLFF+BP (P-MLFF+BP) model is proposed purposely for the case when the number of classes are one hundred. Experimental results show the proposed model yields highest classification accuracy when compared to Kwok's model [8] and a simple MLFF+BP neural network.

Due to the fact that the performance of a neural-based classifier is affected with selection of appropriate textual descriptors, we also propose two different descriptor extraction methods in this paper. One is keyword extending

process in order to reduce the total number of unique descriptors which are used as a classifier's input textual vector. The other is appending process in order to find the related descriptors and improve the similarity of texts in the same class. Experimental results show that the proposed approaches indeed yield higher accuracy when they are implemented to neural-based classification models

In following sections, textual classification with different neural-based models, namely, Kwok's model, MLFF+BP, and P-MLFF+BP are described in Section 2. Two descriptor extraction strategies, keyword extending and descriptor appending, are proposed and the experiments are analyzed in Section 3. Section 4 is the conclusion.

2. NEURAL-BASED TEXTUAL CLASSIFICATION

Textual classification model based on neural networks shown in Figure 1 has one processing for training state and another for testing state. In training state, when a text arrives, textual processing extracts its descriptors and stores them into descriptor statistical information. Each text will be represented by a list of weighted descriptors in textual representation. These lists of descriptors will be the input of proposed neural network model during classification. If the computed output is not a target class, then network parameters will be adjusted by its learning algorithm.

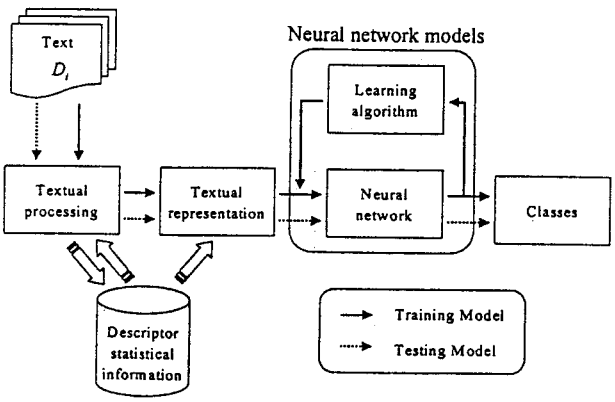

Fig. 1. The flowchart of textual classification model based on neural networks.

2.1 Kwok's Model

Kwok [8] proposed a network with self-learning approach and it was based on probabilistic classification model. This model is very similar to a three-layer feed-forward neural network and can be treated as a semi-neural network. The weights of network links are initialized by statistical probabilistic function. This model also uses a different step function as activation function. The network architecture is shown in Figure 2.

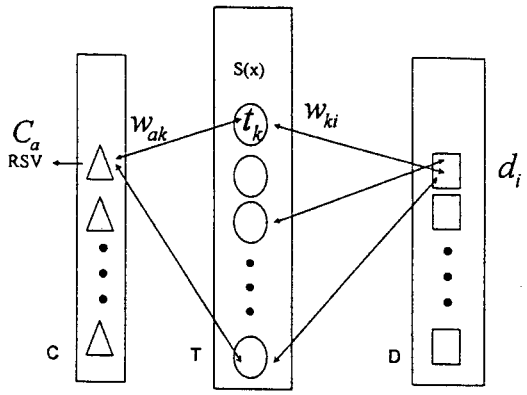

Fig. 2. Three layers self-learning network approach [8].

In the initial stage, the weights of w_{ak} and w_{ki} are initialized by the predefined probabilistic functions. S(x) is a activation function. During self-learning stage, the maximum output value (RSV) of document d_i will be computed for all documents. The RSV of the desired class C_a will be increased by adjusting the weight w_{ak} for each iteration. Self-learning process stops only when the RSV of desired class C_a is reached to a maximum for all documents.

2.2 MLFF+BP Neural Network

MLFF+BP is a supervised neural network as shown in Figure 3. When an input vector is present to this network, the computed output will be compared with target output and an error feedback will be determined. This error feedback will be the input of back-propagation (BP) training procedure.

The weights of MLFF+BP are initialized randomly and adjusted by BP training procedure. In MLFF+BP network, the error (energy) function is given by Equation (1) [5]:

$$E = (1/2)\sum_j (T_j - O_j)^2 , \quad (1)$$

where T_j is the computed output and O_j is the target output. Error function E can be minimized by the partial differential equation as Equation (2):

$$\Delta W_{ij} = -\eta \cdot \frac{\partial E}{\partial W_{ij}} , \quad (2)$$

where η is learning rate. If activation function is the sigmoid function then the final formula for updating weights matrix W_{ij}^{old} to be W_{ij}^{new} will be like Equation (3):

$$W_{ij}^{new} = W_{ij}^{old} + \Delta W_{ij} , \quad (3)$$

where ΔW_{ij} is computed as Equation (4):

$$\Delta W_{ij} = \eta \cdot \delta_j^q \cdot O_j^{q-1}, \tag{4}$$

and δ_j^q is computed as Equation (5):

$$\delta_j^q = \begin{cases} (T_j - O_j) \cdot O_j \cdot (1 - O_j) & \text{,if } q \text{ is output layer} \\ \left(\sum_k \delta_k^{q+1} \cdot W_{kj}\right) \cdot h_j \cdot (1 - h_j) & \text{, otherwise.} \end{cases} \tag{5}$$

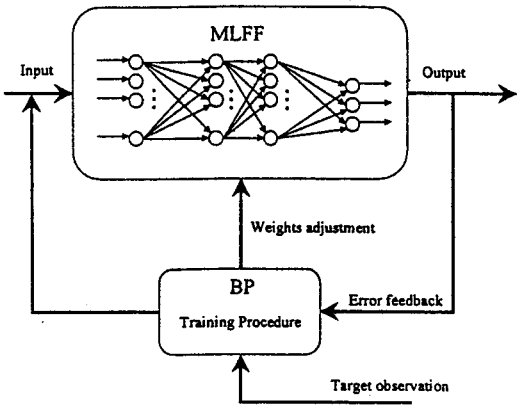

Fig. 3. MLFF+BP neural network [15].

Under supervised learning strategy, BP algorithm is based on gradient descent rule and weight matrix is adjusted by error feedback. This error E can be determined by comparing computed output with target output and be minimized.

There are two main advantages of MLFF+BP neural network model. One is the capability of parallel processing as other neural networks, that is, all elements in the input or output vectors with real values are computed in parallel. The other is the design of hidden layers which can improve the learning capability of MLFF+BP.

In fact, a MLFF+BP without hidden layer does not have any learning capability. Its capability can be increased only with one or two hidden layers; it cannot be increased with more than two hidden layers [5]. Hence the following proposed P-MLFF+BP model is based on a MLFF+BP with one hidden layer.

2.3 P-MLFF+BP Neural Network Model

The proposed P-MLFF+BP model as shown in Figure 4 is composed of k's (i.e. $N_1,\ldots,N_i,\ldots,N_k$) three-layer MLFF+BP networks for k's classes. Each network N_i has only one output value with interval [0,1] and one hidden layer. The output value C_i can be seen as the similarity value that reflects the relationship between a text and class i.

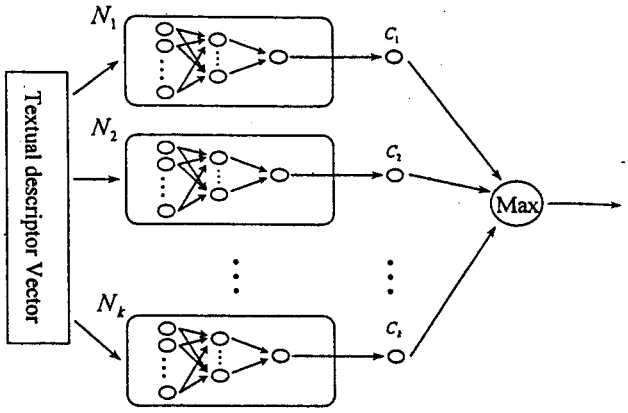

Fig. 4. The Parallel-MLFF+BP neural network.

In P-MLFF+BPE model, each text will be represented with a textual descriptor vector whose dimension is the total number of unique descriptors in a corpus and will be input to each network N_i in parallel. Finally, a maximum selector (MAX) is used to decide the final class for an input text. Since each network N_i learns only one class, the overfitting problem with a simple neural-based classifier will be reduced at the expense of training time.

3. EXPERIMENTS AND ANALYSIS

The experimental flowchart is shown in Figure 5 in which the textual corpus are extracted from National Chiao-Tung University Online Chinese Textual Database (http://ovid.infospring.nctu.edu.tw). We collect total 20000 thesis abstracts as well as the corresponding titles and user-keywords. User-keywords collection collects keywords given by users. Descriptor correction will correct typing errors and treat complex and simplified Chinese characters to be the same. Keyword extending process extends a long user-keyword to short basic grams as text descriptors. Descriptor appending process appends those related descriptors extracted from textual abstracts and titles in order to improve textual similarity in the same class. Finally, each descriptor will be weighted by descriptor weight assignment and a list of weighted descriptors is represented with a vector and is present to the proposed classifier.

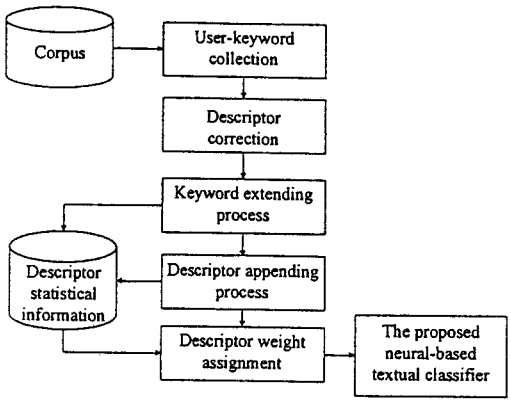

Fig. 5. The experimental flowchart.

3.1 Keyword Extending Process

There are one to twelve Chinese user-keywords per text and 43071 unique user-keywords for total 20000 texts. When using such number of unique user-keywords as the input to a neural-based classifier, it will make training process difficult for present hardware. In order to reduce the dimension of input vector the proposed keyword extending process will cut longer user-keywords with four-up Chinese characters into bigrams or trigrams which are used as textual descriptors. Following is the Cutting Rule employed in keyword extending process:

Long-Keyword Cutting Rule

Assume that $S = \{s_1, s_2, \ldots, s_L\}$ is an order set and
s_l $\{1 < l < L\}$ is a Chinese character,
then
$S' = \{P_1, P_2, \ldots, P_m\}$ is an extended set, if
P_i is an partial order set of S for $i = 1, \ldots, m$ and,
$\min\_length \le |P_i| \le \max\_length$,
$P_1 \cup P_2 \cup \ldots \cup P_m = S$.

There may produce more than one extended set by the rule, the set selection is based on descriptor occurrence frequencies. For example, S = {'資', '料', '庫', '系', '統'} and (min_length, max_length) = (2, 3), then S′ ={ '資料', '庫系統' } or { '資料庫', '系統' }. If frequency('資料庫') ≥ frequency('庫系統') then S′ = { '資料庫', '系統' }.

Originally each text has 5.44 user-keywords in average and each keyword contains two to twenty-three Chinese characters. After keyword extending process, the total number of unique keywords is almost half reduced (shown in Table 1). Meanwhile, the number of descriptors per text will be increased from 5.44 to 7.93 and that will improve the similarity among texts with the same class.

	Total unique descriptor	One occurrence	More occurrence	Ave number of descriptors per text
Original	43017	34416 (80%)	8601 (20%)	5.44
Extending	25030	14963 (59.7%)	10067 (40.3%)	7.93
Appending	24809	9157 (36.9%)	15652 (63.1%)	10

Tab. 1. Frequency distribution.

3.2 Descriptor Appending Process

Since the number of descriptors in input vector will affect textual classification and there is a large variation of descriptor distribution as shown in Figure 6 (from one to thirty-three descriptors per text), each text will be represented with the same number (namely 'ten') of descriptors in our experiments. In the corpus, there are 73.5% texts whose number of descriptors is less than ten after keyword extending process, so descriptor appending process is purposely designed to extract those related descriptors from thesis title and abstracts.

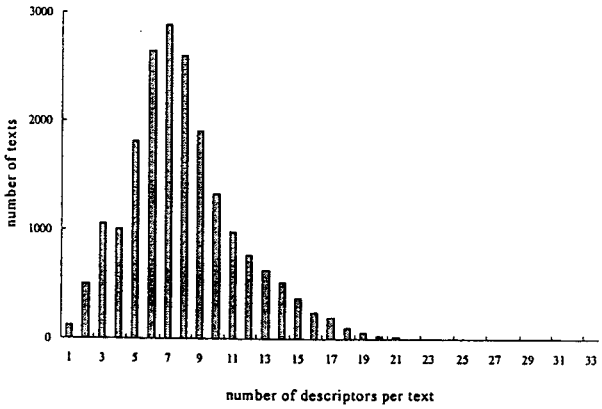

Fig. 6. Descriptor distribution after keyword extending process.

Finding the related descriptors with respect to user-keywords is based on the following similarity function.

$$sim(g_i, g_j) = \frac{2(P_{g_i,g_j})}{P_{g_i} + P_{g_j}}, \qquad (6)$$

where

P_{g_i} : the number of texts includes descriptor g_i,

P_{g_j} : the number of texts includes descriptor g_j,

P_{g_i,g_j} : the number of texts includes both g_i and g_j.

Each related descriptor will be appended to the end of user-given descriptor list in the order of the similarity value. As a result, the percentage of textual descriptors which occur once in the corpus is furtherly decreased from 59.7% to 36.9%. Consequently, the similarity value among texts in the same class is increased.

3.3 Descriptor Weight Assignment

After appending process, each ten descriptors will be weighted by the following function:

$$w_{ik} = \mathrm{Nor}\left(freq_{ik} \cdot \left(\frac{N}{docfreq_k} + 1\right)\right), \quad (7)$$

where

w_{ik} : weight of descriptor k in text d_i,
$freq_{ik}$: the occurrence frequency of descriptor k in text d_i,
$docfreq_k$: number of texts containing descriptor k in corpus,
N : number of texts,
Nor(.) : normalization function.

and Nor(.) normalizes the real interval $[0,\infty]$ into $[0,1]$ as Equation (8):

$$\mathrm{Nor}(x) = 1 - \frac{1+z}{(x+1)^z + z}, \quad (8)$$

where x is real in $[0,\infty]$ and z is a normalization factor.

During weighting design, those descriptors extended from user-keywords will have higher weight than those related descriptors. If a descriptor is a user-keyword then its weight will be computed directly by Equation (7). If it is a related descriptor then its weight will be the product of its weight computed by Equation (7) and the weight of its extended user-keyword.

3.4 Model Comparisons

In this paper, models are compared in terms of classification accuracy defined as Equation (9):

$$accuracy = \frac{\text{number of texts correctly classified}}{\text{number of texts in corpus}}. \quad (9)$$

In order to verify that the proposed P-MLFF+BP model is able to reduce the overfitting problem, comparisons between this model and a simple MLFF+BP model are made. One can find that P-MLFF+BP indeed outperforms MLFF+BP with respect to different number of classes as shown in Figure 7. Furthermore the decreasing slope of P-MLFF+BP is less than that of MLFF+BP when number of classes increases.

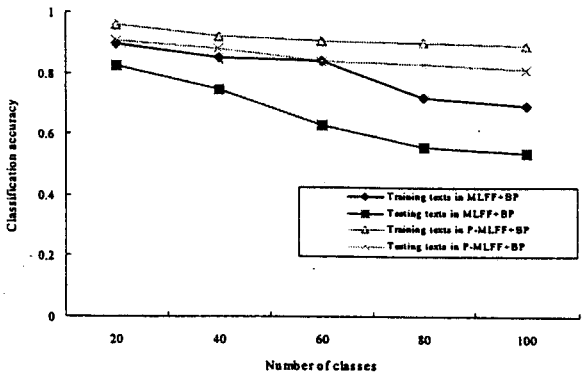

Fig. 7. Classification accuracy w.r.t. different numbers of classes.

Figures 8 and 9 show that the designed descriptor appending process indeed yields higher classification accuracy than keyword extending process when they are applied to different models and the proposed P-MLFF+BP model has the highest classification accuracy. Figure 10 shows that the classification accuracy of P-MLFF+BP model with appending process is higher than P-MLFF+PP with extending process in both weighted and unweighted descriptor cases.

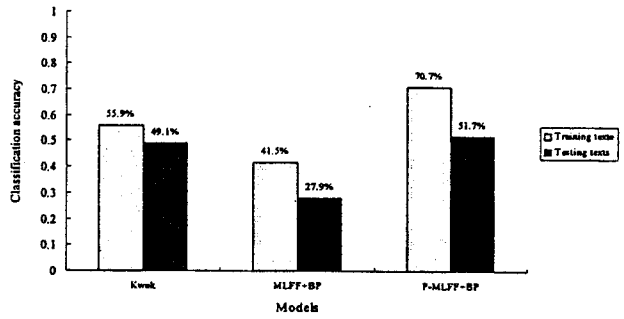

Fig. 8. Classification accuracy of three models after keyword extending process.

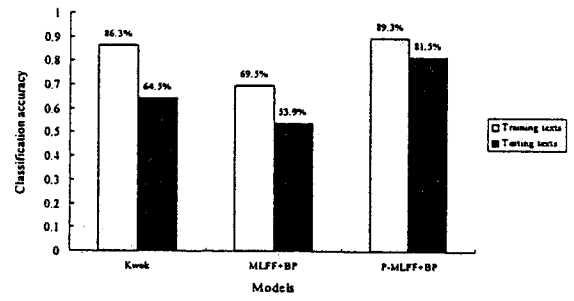

Fig. 9. Classification accuracy of three models after descriptor appending process.

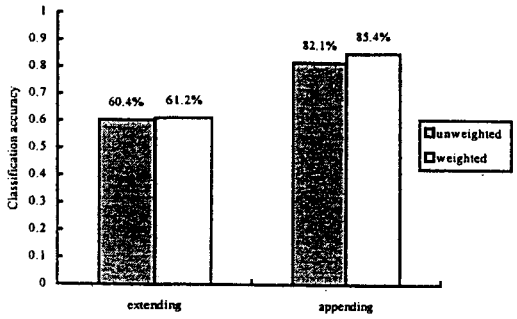

Fig. 10. Extending vs. appending implemented with a P-MLFF+BP.

4. CONCLUSION

In this paper we investigated the feasibility of applying neural networks to textual classification. As an important role of a neural-based classifier, textual descriptors are extracted with two different strategies. One is keyword extending processing which results in reduction of the number of unique keywords and the other is appending process which improve the similarity of the texts in the same class. Experimental results show that both of these strategies indeed improve classification accuracy and speedup training time in terms of reduction of input vector dimension.

On the other hand a parallel neural-based classifier is proposed in order to improve the accuracy when it has to deal with many number of classes during textual classification. Experimental results show the proposed model yields highest classification accuracy when compared to Kwok's model and a simple MLFF+BP neural network.

5. REFERENCES

[1] 洪文斌 (Horng)、黃連進。"使用類神經網路來作文件自動分類之研究，"一九九八分散式系統研討會，成功大學，1998。

[2] H. Broko and M. Bernick., "Automatic Document Classification," Journal of the ACM, Vol. 10, No. 1, pp. 151-162, 1963.

[3] W. B. Croft., "A Comparison of Cosine Correlation and the Modified Probabilistic Model," Information Technology, Vol. 3, No. 2, pp. 113-114, 1984.

[4] W. B. Croft., "A Comparison of Text Retrieval Models," The Computer Journal, Vol. 35, No. 3, pp. 279-290, 1992.

[5] D. W. Patterson. "Artificial Neural Networks: Theory and Applications," MA: Prentice Hall, 1995.

[6] A. Griffiths, L. A. Robinson, and P. Willett., "Hierarchical Clustering Methods for Automatic Document Classification," Journal of Documentation, Vol. 40, No. 3, pp.175-205, 1984.

[7] K. L. Kwok., "Experiments with a Component Theory of Probabilistic Information Retrieval Based on Single Terms as Document Components," ACM Transactions on Information Systems, Vol. 8, No. 4, pp. 363-386, 1990.

[8] K. L. Kwok., "A Network Approach to Probabilistic Information Retrieval," ACM Transactions on Information Systems, Vol. 13, No. 3, pp. 324-353, 1995.

[9] M. E. Maron and J. L. Kuhns., "On Relevance, Probabilistic Indexing, and Information Retrieval," Journal of the ACM, Vol. 7, No. 3, pp. 216-244, 1960.

[10] M. E. Maron., "Automatic Indexing: An Experimental Inquiry," Journal of the ACM, Vol. 8, pp. 404-417, 1961.

[11] F. Murtagh., "A Survey of Recent Advances in Hierarchical Clustering Algorithms," The Computer Journal, Vol. 26, No. 4, pp. 354-360, 1982.

[12] H. T. Ng, W. B. Goh, and K. L. Low., "Feature Selection, Proceptron Learning, and a Usability Case Study for Text Categorization," Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, pp. 67-73, 1997.

[13] J. J. Rocchio., "Relevance Feedback in Information Retrieval: The Smart System-Experiments in Automatic Document Processing," ed. Salton, G., MA: Prentice-Hall, pp. 313-323, 1971.

[14] G. Salton., "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer,", MA: Addison Wesley, 1989.

[15] R. J. Schalkoff., "Artificial Neural Networks," MA: McGraw-Hill, 1997.

[16] Y. Yang., "Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval," Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Ireland, pp. 13-22, 1994.

[17] C. T. Yu, W. S. Luk, and T. Y. Cheung., "A Statistical Model for Relevance Feedback in Information," Journal of the ACM, Vol. 23, No. 2, pp. 273-286, 1976.